DSC 461 Project Report - Job Application Database

Team: Fastest HK Journalist (# 11)

Members: Zhou Xu (zxu17), Kefu Zhu (kzhu6), Hao Jiang (hjiang23)

I. Project Description

Problem Statement

With the proliferation of online job application, many companies suffer from the inefficiency of traditional databases, mostly due to poor performance of spreadsheet. For instance, the application status cannot be updated in a timely manner or contradictory entries were input from multiple ends. These problems increase the time cost and decrease company operation efficiency. The relational database system that we designed in this project bring a solution to manage complex data of job applications for a company and provides a user-friendly platform for HR and applicants to insert, retrieve and modify their profile data. In addition, the database will present information for advertisement platforms that each job is posted.

Target Users

Users of this database include employees (especially HR) of the company as well as job applicants and headhunters who are looking for jobs. To be specific, the administration site will be used internally by HR and the applicant site will be used externally for everyone to view and apply to the positions that are currently open in the company. The administrator of the database will be the data engineer in the IT department who will be in charge of maintaining the database in a daily basis.

List of Relations in The Database

OFFICE

1. ID: The identification number for each office
2. StreetAddress: The street address of the office
3. City: the city name where the office is located
4. State: the two-letter abbreviated codes for the states
5. ZipCode: the 5-digit zip code of the office

Note:

1. The primary key of this relation is ID
2. The StreetAddress has UNIQUE constraint

```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| ID            | int(11)     | NO   | PRI | NULL    |       |
| StreetAddress | varchar(50) | NO   | UNI | NULL    |       |
| City          | varchar(25) | NO   |     | NULL    |       |
| State         | char(2)     | NO   |     | NULL    |       |
| ZipCode       | int(5)      | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 1: Relation Schema for OFFICE

DEPARTMENT

1. ID: The identification number of department
2. Name: The name of department
3. OfficeID: The identification number of office where the department belongs

Note:

1. The primary key of this relation is (ID, Dnumber)
2. OfficeID is a foreign key that references to the ID in OFFICE relation

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ID       | int(11)     | NO   | PRI | 0       |       |
| OfficeID | int(11)     | YES  | MUL | NULL    |       |
| Name     | varchar(50) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 2: Relation Schema for DEPARTMENT

RECRUITER

1. ID: The identification number of employee (recruiter)
2. FirstName: The first name of recruiter
3. LastName: The last name of recruiter
4. Position: The position of employee
5. DepartmentID: The ID of department where recruiter belongs

Note:

1. The primary key of this relation is ID
2. DepartmentID is a foreign key that references to the ID in DEPARTMENT relation

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(30) | NO   | PRI | NULL    |       |
| FirstName    | varchar(30) | NO   |     | NULL    |       |
| LastName     | varchar(30) | NO   |     | NULL    |       |
| Position     | varchar(50) | NO   |     | NULL    |       |
| DepartmentID | int(11)     | YES  | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 3: Relation Schema for RECRUITER

POSITION_

1. ID: The identification number of open position
2. DepartmentID: The identification number of department
3. Type: The type of open position (FullTime/PartTime). Default value is FullTime
4. PostDate: The date when position opens to application
5. Title: The title name of open position
6. Salary: The annual salary for open position
7. VisaSponsorship: Whether the open position sponsor work visa (YES/NO). Default value is YES

Note:

1. The primary key of this relation is ID
2. DepartmentID is a foreign key that references to the ID in DEPARTMENT relation

```
+-----------------+-------------+------+-----+----------+-------+
| ID              | int(11)     | NO   | PRI | NULL     |       |
| Type            | char(8)     | YES  |     | FullTime |       |
| Title           | varchar(50) | NO   |     | NULL     |       |
| DepartmentID    | int(11)     | YES  | MUL | NULL     |       |
| RecruiterID     | varchar(30) | YES  | MUL | NULL     |       |
| PostDate        | date        | YES  |     | NULL     |       |
| Salary          | int(11)     | YES  |     | NULL     |       |
| VisaSponsorship | varchar(15) | YES  |     | Yes      |       |
+-----------------+-------------+------+-----+----------+-------+
8 rows in set (0.00 sec)
```

Figure 4: Relation Schema for POSITION_

APPLICANT

1. ID: The auto incremented identification number of applicant
2. Applicant_ID: The office identification number of applicant (e.g. A10001)
3. FirstName: The first name of the applicant
4. LastName: The last name of the applicant
5. Email: The contact email address of the applicant
6. School: The name of school where the applicant graduated
7. Degree: The highest degree obtained by the applicant
8. Major: The major of the applicant

Note:

1. The primary key of this relation is ID

```
+--------------+-------------+------+-----+---------+----------------+
| ID           | int(11)     | NO   | PRI | NULL    | auto_increment |
| Applicant_ID | varchar(30) | YES  | UNI | NULL    |                |
| Email        | varchar(50) | NO   |     | NULL    |                |
| FirstName    | varchar(30) | NO   |     | NULL    |                |
| LastName     | varchar(30) | NO   |     | NULL    |                |
| School       | varchar(50) | YES  |     | NULL    |                |
| Degree       | varchar(30) | YES  |     | NULL    |                |
| Major        | varchar(30) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Figure 5: Relation Schema for APPLICANT

APPLY_TO

1. ApplicantID: The office identification number of applicant
2. PositionID: The identification number of position that the applicant applied to
3. AdsPlatform: The name of platform from which the applicant saw the open position

Note:

1. The primary key of this relation is (ApplicantID, PositionID)
2. ApplicantID is a foreign key that references to the ID in APPLICANT relation
3. PositionID is a foreign key that references to the ID in POSITION_ relation

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| ApplicantID | varchar(30) | NO   | PRI |         |       |
| PositionID  | int(11)     | NO   | PRI | 0       |       |
| AdsPlatform | varchar(50) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 6: Relation Schema for APPLY_TO

ADSPLATFORM

1. PositionID: The identification number of position
2. AdsPlatform: The name of platform on which the position was advertised

Note:

1. The primary key of this relation is (PositionID, AdsPlatform)
2. PositionID is a foreign key that references to the ID in POSITION_ relation

```
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| PositionID  | int(11)     | NO   | PRI | 0       |       |
| AdsPlatform | varchar(50) | NO   | PRI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figure 7: Relation Schema for ADSPLATFORM

Web Interface

The final web interface of our administration site (for internal use, eg. HR or website administrators) is shown below in Figure 8. This is one example from our office.php site: (http://betaweb.csug.rochester.edu/~zxu17/office.php). There are 3 parts: header (1), control panel (2) and table (3). In the header, users can switch between different relations by clicking the links in header. In the control panel, users can perform actions on the database including insertion, deletion, search and reset. The data in the corresponding table will be displayed in the table region, where users can either inspect the data in the relation or keep track of the modified or queried data after performing actions.

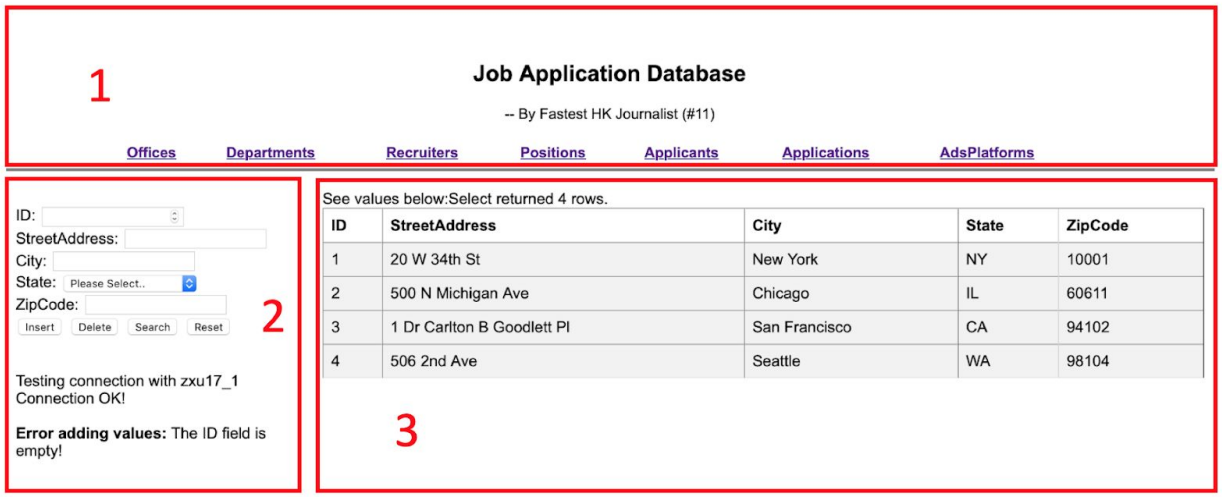

Figure 8: Elements of the administration website

In the control panel, as shown in Figure 9, it can generally be splitted into 4 regions: input (1), actions (2), connection status (3) and messages (4). To use the insert function, users need to input valid data into the corresponding fields and click the insert button in the action region. If the action is successful, a message will be displayed in the message region, telling the user that the value is added and the inserted data will be shown in the table; if something goes wrong with the action, the error message will be displayed telling the user what the error is. The delete and the search functions work similarly to the insert function that the users have to enter exact values in the input fields (can be one or multiple fields), then the database will perform corresponding delete or search queries. After searching, the users can click on the reset button to go back to the original table. For all of the insert, delete and search functions, an equivalent SQL query will also be displayed to give users a general idea of what they are doing. This would be particularly useful if the users know SQL.

Figure 9: Elements of the control panel

The external website for applicants (http://betaweb.csug.rochester.edu/~zxu17/index.php) is shown below in Figure 10. It has three regions, similar to the administration site. However, since it is for external use, the users can only view partial information related to the open positions. If they want to apply for the jobs, they have to enter their information in the second form in the control panel region. If their information is submitted successfully, there will be a message shown telling them the company has received the application; if something goes wrong, the applicants will get relevant error messages telling the source of the error and what steps they should take next.

Figure 10: Elements of the application website

When designing the websites, we are aware that they do not fit perfectly into real-world scenarios, so we have made some assumptions and rules when planning for the functionalities of the websites. First, we assume our administrators conform to the company rules and they will most likely input only valid information without maleficent purposes. Even though we have some protection mechanisms, for example, preventing them from modifying the database other than insert, delete and search functions, they still have freedom to insert dirty data or delete useful data. So, we need to define company rules than explicitly implement some extra validation processes in our database. Second, auto incrementing ID is handy, but here we choose to give administrators freedom to insert customizable data including the ID, so we keep the ID a necessary field to fill when inserting data. However, this does not apply to the applicants relation, because when applicants send their applications, we would like to have the ApplicantID auto generated and stored in the database. In addition, we did not make the user login function because we make the assumption that the applicants cannot modify their sent applications after they submission. On the other hand, login is not required for the administrator site because we assume anyone with the internal access is able to view and modify our database.

Source of Data

The relations are populated using data created by ourselves while referencing to real-world data to some degree. In the future, bulk data can be inserted directly into the database through MySQL server or from the "upload bulk data" function (will be implemented, see future work). Other than that, single pieces of data can be easily inserted through the website applications, which is already well established now.

II. Contribution

Project Milestone	Name	Contribution	Contribution (%)
1	Hao Jiang	Propose Database Write Proposal	33%
1	Kefu Zhu	Propose Database Write SQL code Write Proposal	33%
1	Zhou Xu	Propose Database Write Proposal	34%

2	Hao Jiang	Draw ER diagram and web interface Write Report	33%
2	Kefu Zhu	ER Transformation Write SQL code Write Report	33%
2	Zhou Xu	Propose ER ER Transformation Write Report	34%
3	Hao Jiang	Write php/html code	33%
3	Kefu Zhu	Write SQL code Write php/html code	33%
3	Zhou Xu	Write php/html code	34%
4	Hao Jiang	Design web interface Write Report	33%
4	Kefu Zhu	Write Report	33%
4	Zhou Xu	Design web interface Write php/html code Write Report	34%

III. Summary

The database and its web interface we created in this project can successfully function with basic user interface and necessary functionalities including data insertion, deletion and search. Users can access the web interface and edit the database to certain degree with corresponding permission levels. The web interface is able to process the POST requests sent by the user and transforms it into SQL query to interact with underline database.

Furthermore, we added some additional constraints to prevent unintentional typos as well as potential sabotage actions. For example, we leveraged regular expression to evaluate input values before sending them to the SQL execution stage, including specifying the correct input format (e.g. date format in the post date of position) as well as restricting the use of unnecessary special characters that could potentially be exploited to destroy the database. However, we do believe there are better alternatives for such problems that should be investigated in the future.

The most challenging part of the project for us is the implementation of php and html code. Since none of us are familiar with php and html, we spent a great portion of our time on debugging the code. Nevertheless, the biggest gain from this project for us is to practice the database and SQL knowledge by building a database with real-world application and push it online.

IV. Future Work

The current database is still restricted in uses by its limited functionalities. Due to the limited time for this project, we were not able to implement every feature we came up with. Below are a list of improvements we believe that can be done in the future:

User Interface

1. The tables can be splitted and displayed in multiple pages to save time from scrolling. The number of records shown in each page can be added at the bottom of the page and it should able to be customized by the user.
2. The web interface should supports insertion by uploading bulk data file.
3. The current search feature does not support fuzzy search, which is a useful functionality and should be implemented.
4. Simple visualizations about the data can be implemented for faster digestion of information within the data. For instance, a pie chart of AdsPlatform attribute in the Adsplatform page can be shown to the HR to better understand the big picture.

Database

1. The current version of MySQL on betaweb does not support “ON DELETE SET DEFAULT”. There should be an alternative solution.